Semi-Automatic Stencil Creation through Error Minimization

Jonathan Bronson*
University of Maryland
Baltimore County

Penny Rheingans†
University of Maryland
Baltimore County

Marc Olano‡
University of Maryland
Baltimore County

Figure 1: An example of input mesh, minimum error bridges, and final stencil image.

Abstract

Creating physical stencils from 3D meshes is a unique rendering challenge that has not been previously addressed. The task is a problem of two competing goals: forming a single, well-connected and stable stencil sheet, while simultaneously limiting the error introduced by pieces of bridging material. Under these conflicting goals, it can often be difficult to create visually pleasing stencils from complicated imagery by hand. Even for well-behaved images, expressive stencils can be time-consuming to craft manually.

We present a method for generating expressive stencils from polygonal meshes or images. In our system, users provide input geometry and can adjust desired view, lighting conditions, line thickness, and bridge preferences to achieve their final desired stencil. The stencil creation algorithm makes use of multiple metrics to measure the appropriateness of connections between unstable stencil regions. These metrics describe local features to help minimize the distortion of the abstracted image caused by stabilizing bridges. The algorithm also uses local statistics to choose a best fit connection that maintains both structural integrity and local shape information. We demonstrate our algorithm on physical media including construction paper and sheet metal.

Keywords: automatic stencil creation, theatrical gobos, non-photorealistic rendering, view-dependent visualization

1 Introduction

Often in graphic design the details of an image are not as important as the essential impression the viewer takes away from it. Stencils capture this essential impression, giving an abstracted view of objects in a way that allows them to be readily identifiable. Additionally, stencils provide an easy means of reproduction for logos, symbols, and images.

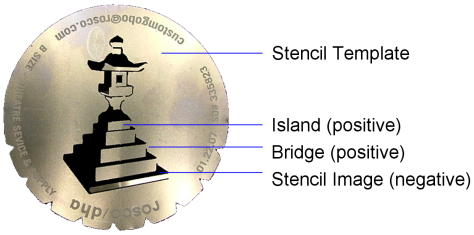


Figure 2: An example of a lighting gobo. In a valid stencil such as this, the template is a single, connected piece of material. All islands are connected by a bridge to the main sheet.

In theatrical lighting, a metal or glass stencil, known as a *gobo* or *template*, is placed before the lens of a lighting instrument, much like a slide in a slide projector. The lighting instrument can then project these images onto the scene. Gobos made of glass avoid the connectivity constraints of stencils, while allowing color within the gobo to be projected. However, metal gobos, such as the one shown in Figure 2, remain the most popular for their price and durability. A similar device known as a *cookie*, used for film or photography, is a stencil cut from plywood or poster board and placed in front of the light.

Stencils, gobos, and cookies all have the common features of being constructed from a single sheet of flat material, and of including structural elements in the design such that the sheet stays together. Stencil media can be either paper, poster board, plastic, wood, or metal, with the thickness of the material defining the level of detail that the stencil can support. A thinner material can capture more detail but is less likely to maintain its shape.

Throughout this paper we will refer to all these forms, regardless of the media or purpose, by the generic term *stencil*. A stencil can be characterized as containing two types of regions, one positive and one negative. A negative region is an empty space through which paint or light may pass, while a positive region is an entirely connected surface of material, either directly connected or connected by unobtrusive bridges through negative space. Images in the pa-

*jonbronson@umbc.edu
†rheingan@umbc.edu
‡olano@umbc.edu

per will represent negative space with black and positive space with white. When used on actual media, the negative space can be any color of paint or projected light, and the positive regions will be a cast shadow or base color of a painted material.

2 Motivation

Until now, artistic stencils have been designed manually, usually from pre-existing pictures or photographs. Artists decide which regions will be cut away, while being careful not to remove important regions of the image. To accomplish this, they must manually identify regions that will fall away when cut out, and designate a strip of material that will remain to maintain connectivity. The pieces that will fall out are commonly referred to as *islands*. Naturally, the connections between islands, or from an island to the stencil frame, are referred to as *bridges*. Figure 2 shows a gobo with several of these features identified.

The challenge for the artist throughout this process is to maintain the physical integrity of the template, while being careful not to remove important image details. This task involves visually identifying all potential islands as well as choosing aesthetically pleasing bridges to connect these islands to the rest of the stencil. Even for somewhat simple stencils it can sometimes be very difficult to visually discern which regions are 'connected' to the stencil. As the complexity increases, this task becomes increasingly vexing. High frequency details can lead to a very large number of islands, each of which must be identified and accounted for by the artist. Image editing software can aid this process, but it is still up to the artist to ensure connectivity constraints.

If a technique were available to help choose bridges appropriately, professional artists would be more free to focus on the imagery itself and others might be more inclined to utilize the advantages of stencils. Furthermore, a mechanism to generate these stencils directly from 3D geometry provides a means of bringing digital artwork back into a real world medium. We present an approach for the semiautomatic creation of physically valid stencils from images or arbitrary polygonal models. Users provide the geometry or prepared 2D images for the stencil creation and our algorithm chooses a set of bridges that causes the least amount of information loss while still fully connecting all disjoint regions. We demonstrate two methods of construction, manually cutting from a printed template and photo-etching, as well as two applications, painting templates and lighting gobos.

3 Related Work

Automatic generation of stencils has not been a widely studied topic, so our method draws upon the work of researchers in other areas of non-photorealistic rendering. We borrow techniques from cartoon shading to achieve the stark dark and light features. Silhouettes [Markosian et al. 1997; Raskar and Cohen 1999] and suggestive contours [DeCarlo et al. 2003] also have a strong effect on the quality of the final stencil. Depending on the exact style wanted, a wide variety of approaches is available [Isenberg et al. 2003]. These chosen methods provide us with a useful abstraction of the image to be created. At a minimum, we must reduce the image to exactly two tones, but the manner in which we do this determines the final style. That being said, if one intends to generate stencils from photographic data a powerful abstraction method will greatly improve the clarity of the stencil image. DeCarlo and Santella [2002] provided one such method that could be used in conjunction with our stencil algorithm to help develop clean stencils from photographs. Their segmentation method might be a starting point for an alternative approach to choosing bridges.

Non-photorealistic rendering (NPR) produces images that mimic the styles artists use in physical media. Several techniques aim at producing stylized images that, like stencils, are at their core a series of segmented regions. Examples include the rendering of batiks [Wyvill et al. 2004] and mosaics [Hausner 2001; Kim and Pellacini 2002]. The key difference between the other techniques and stencil creation is that in the other algorithms a method for segmenting the image in a coherent way is the heart of the algorithm. In contrast, we are starting with a series of segmented regions and looking for a suitable means of reconnecting them. The final product is similar in the sense that the result is a lossy abstraction of the original image, and the artistic style comes from the manner in which that loss occurs. In batik the style comes from the cracks, in mosaic from the gaps between tiles as well as the color abstraction, and in stencils from the artificial bridges connecting islands.

A few works, like ours, have taken the process one step further to convert the NPR results back into the physical media that inspired them. The program and algorithms developed can serve as a means to express inherently mathematical forms in artistic styles [Ferguson 1992; Eisenberg et al. 2005], as a means of creating 3D analogs to 2D art [Yen and Séquin 2001; Raskar et al. 2002; Kaplan and Salesin 2004], or as a means to aid an artist through the tedious or complex portions of the artistic process [Lang 1996; Mitani and Suzuki 2004]. Our work falls in this final class, assisting the artist to produce complex, yet inherently artistic, works.

Perhaps the closest recent work is in topological simplification and filtering. Williams and Rossignac [2005a; 2005b] have created topological filtering operators that can simplify the topology of regions of an image, but their operators were not intended to produce stencil images, and do not. The creation of bridges between positive regions is counter to the goals of their algorithm. Wood et al. [2004] perform similar operations to cut loops and fill holes in 3D isosurface models, but their algorithm operates on surfaces in 3D, not 2D stencils.

4 Approach

During the creation of a stencil, there are two competing pressures driving the bridge selection process. One is the desire to maintain a single, structurally sound sheet of material. It is maximally satisfied when there are no holes in the sheet. The other is the desire to reduce the error caused by the two-toned abstraction of the stencil. It is maximally satisfied when there are absolutely no bridges added. The challenge of stencil creation is finding a balance between these pressures, whether through an algorithm or by hand.

The style of the bridges may also be considered as an added constraint. Typically in the physical media, we see bridges as straight strips of material. One downside to this is that the strip imposes its shape onto the negative region of the stencil that it is crossing. One way to address this problem would be to draw bridges along the principal curvatures. A strong argument has been made that strokes in these directions tend to help the visual system see shapes more clearly [Girshick et al. 2000]. However, a curved bridge will also be longer and introduce a greater change to the overall edge boundaries. The user must decide the trade-off between longer bridges and better shape suggestion.

The general approach taken in this paper is one of abstraction and error minimization. Beginning with a two-toned abstraction, we identify the positive regions of the image that are not connected to the outer frame through some path. We call these disconnected regions *islands*. We will add the minimum number of bridges necessary and in optimal locations to connect all islands into one continuous piece of material. If we consider the islands as nodes in a graph, and all possible bridges as edges between nodes, we are

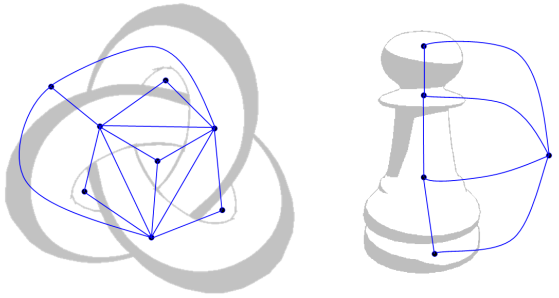


Figure 3: Two images are shown with the graphs that represent them superimposed. Each node represents either an island or the stencil frame itself. Each edge represents a set of possible ways to union the islands.

interested in constructing a minimal spanning tree where cost is defined by how badly a bridge disrupts the abstract image. Figure 3 shows two examples of images with their potential connectivity graph superimposed.

For users who have polygonal data available, the first stage is interactive, allowing the user to tune a two-toned image by adjusting viewing and lighting conditions. Alternatively, the user can provide a two-toned image directly. The second stage is entirely image-based. Constructing the data for bridge optimization takes a few seconds, so the full scene-to-stencil process is not real-time, but it is fast enough that users are free to alter the scene parameters and quickly see the affect on the output stencil.

5 Two-Tone Image Creation

Part of the power of a stencil is its incredible abstraction ability. We can remove much of the lighting information and still clearly discern the object. We use a combination of silhouette line drawings [Markosian et al. 1997] and a simple toon shader for creating black and white images from polygonal models. The shader starts with a Lambertian model of lighting with specular highlights and a user-defined threshold. If the intensity of a given pixel is above the threshold, it is made a positive (white) region; if it is below, it becomes a negative (black) region. It is conceivable that an artist would like a stencil of the highlighted regions of a scene, rather than the shaded ones. It is a trivial matter to reverse the stencil process and connect lit regions, so we only present the first case.

The lighting source chosen for the scene has a large impact on the resulting two-toned image. It is characteristic of 20th century artistic stencils to be created in a chiaroscuro shading style, that is, with dramatic lighting from some angle. For this reason, we chose to implement the lighting as a point light source at close proximity to the geometry. Other choices for lighting will not affect the stencil creation process, but may greatly impact the resulting aesthetic value. For very simple objects, this two-toned rendering will occasionally create an image that is itself a valid stencil, but in general we need to do more.

It is usually not desirable to have edges entirely removed in light regions. For this reason, we perform silhouette rendering [Markosian et al. 1997] and add suggestive contours [DeCarlo et al. 2003] to increase the level of detail in some models not easily identifiable strictly from the shadow regions. Excessive contours are avoided since stencils are meant to be an abstraction. The results of this can be seen in Figure 4. The left image shows the model rendered with standard Lambertian model. The center image is the same

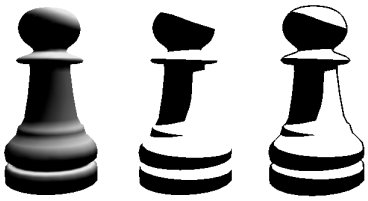


Figure 4: This image conveys the simple process to create a two-toned image from the original rendering.

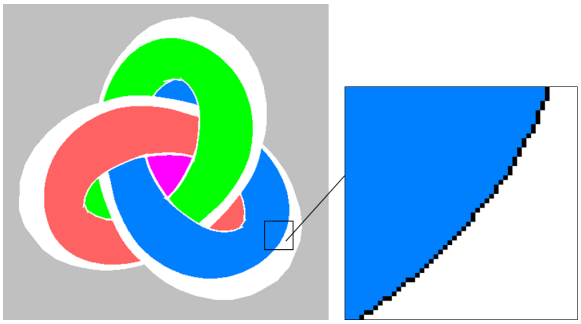


Figure 5: Each island is identified and all border pixels are stored as possible bridge starting locations. The right close-up shows candidate locations darkened for clarity.

model rendered strictly with the two-tone thresholding, while the right image is the two-toned shader with silhouettes added. Notice that all shape information is lost within both dark and light regions and must be implied. The most important features added by this approach occur at the regions where positive and negative meet. These shadow lines tend to accentuate the curvature of the object while the silhouettes give a definitive boundary to areas that might otherwise be unclear.

6 Island & Candidate Identification

Having reduced the image down to only two values, the next task is to identify regions that must be connected. We identify islands with a simple fill algorithm and store them in an island data structure so that we can later treat them as nodes in a graph. Figure 3 shows two examples of connectivity graphs. All border pixels within an island are considered candidate locations for the start of possible bridges.

Since we are working in black-and-white image space, we may have issues with aliasing. An orphaned, positive pixel completely surrounded by a negative region has no orientation information. The original renderer places this pixel to approximate specific color and shading. Unfortunately, adding a bridge to such a small island greatly overcompensates for the importance of the pixel itself. To avoid this, we implement a topological filtering [Williams and Rossignac 2005a] composed of dilation and erosion to close off these small areas before the stencil process begins. As with other forms of aliasing, supersampling can help alleviate the problem.

7 Bridge Selection

Once we have identified all islands we must choose a method to connect them. After considering some of the implications of the bridging process, a solution presents itself. Clearly all islands require at least one bridge to be connected to the stencil sheet. Fur-

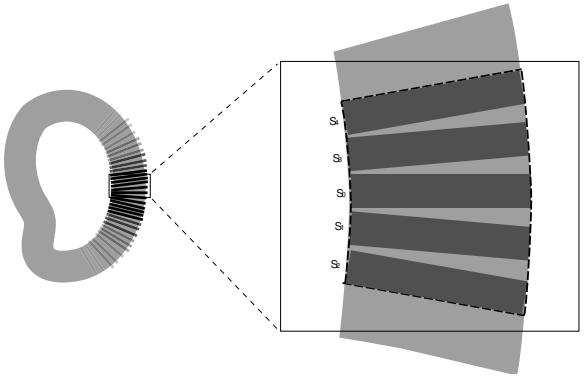


Figure 6: Bridge costs can be approximated by a series of small strips. Joining together adjacent strips will create wider bridges. Once a strip, S_0, is chosen as the new starting location for a bridge, the least costing adjacent strip S_i will be added into the bridge set until the required width is achieved. Once all strips S_0 through S_N have been identified, a bridge polygon can be formed from their endpoints.

ther, since at least one bridge choice is required, minimizing the cost of this bridge will reduce the overall error caused by the abstraction. This process of connecting minimal error bridges until the stencil is fully connected is a slightly more complicated minimal spanning tree problem. Instead of a single edge between nodes, we are forced to consider a set of edges between any pair of nodes. Each edge within this set is really a candidate bridge between the two stencil regions represented by the nodes. The abstract edge in the stencil graph will be given the cost of the minimal costing bridge within its set. However, for any given island there are uncountably many bridge choices and this set cannot be exhausted. Instead, we break up the perimeter of the island into very small bridge strips. Using this approach, a bridge of any given width may be formed as the aggregate of several neighboring strips. Similarly, the cost of a given bridge will simply be the combined cost of the strips from which it is comprised.

As seen in Figure 6, these bridge strips are simple paths in 2D space, originating from a candidate on one island and finishing on a different island. All pixels that the path crosses through will be considered part of the strip. The only consistent property of these path directions in the physical media is that they tend to connect at very orthogonal angles. Our method is to trace a path in the direction of the image gradient. A short walk will lead us to a neighboring island and we simply record pixels crossed along the way. We need not worry about concave islands because our bridge strips only terminate when they have reached a different island than they originated from. Excessively long paths will incur too large a cost to ever be chosen. In order to identify when a strip has bridged a negative space, we store a sparse mapping of pixels to islands which contain them. Each island in the image will build a set of these strips that will completely encompass the perimeter. One or more groups of these strips will later combine to form the full bridge in the final image.

7.1 Bridge Strip Cost Function

In order to use these bridge strips as approximators, we need a means to evaluate their cost. This cost will be the total sum of the error introduced by each new pixel within a particular strip, with penalties for bridges which are in poor locations. The following function C represents our combined cost for a particular bridge

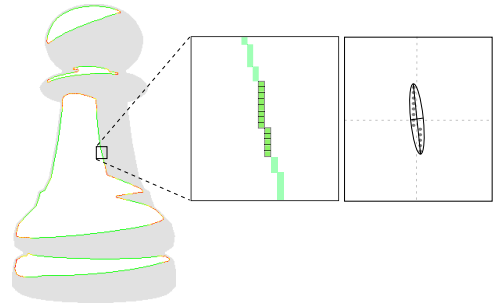


Figure 7: This image shows the curvature values calculated for candidates. The scale ranges from red (very curvy) to green (very straight). To the right is the ellipse generated by the eigenvectors of the covariance matrix of the local window.

strip. The cost function consists of two types of information: *static costs*, which are the accumulation of individual pixel costs; and *semantic costs*, which are formed by the location and orientation of the strips and the fitness of their local neighborhood. Given a set of P pixels which comprise the strip, we write the cost function as:

$$C(P) = w_0 \sum_{i=1}^{P} I_i + w_1 L^{\alpha} + w_2 \kappa + w_3 \sigma_C^2 \tag{1}$$

where

$\sum_{i=1}^{P} I_i$ = Combined Intensity Difference

L^{α} = Bridge Length Penalty

κ = Local Curvature of Island border

σ_C^2 = Variance of Cost over Local Neighborhood.

and $w_0 - w_3$ are the corresponding weights.

There are three basic components that contribute to the cost of the bridge strip. First, each pixel will alter the intensity of the original two-toned image. This error can be accumulated along the path of the strip by storing the squared difference of intensity between the original grayscale rendering and pure white, the color of the bridges. Although this value is accumulated across multiple pixels, it is not sufficiently indicative of the aesthetic cost of longer bridges. To account for the bridge length in and of itself, we add a separate penalty, L^{α}. This penalty should be very minimal at short distances, but grow quickly as bridges become longer. The reason for this is that at small distances the other metrics are much more important factors in choosing aesthetically pleasing bridges. However, in practice, as bridges become longer, the length quickly outweighs the cost of bad lighting and curvature values. We found $1 < \alpha < 2$ to be sufficient for this purpose.

We would also like to avoid placing bridges in regions of high curvature. These areas are usually essential to understanding the shape of the object. We account for this by computing the covariance matrix of the coordinates of neighboring candidate bridge locations. This covariance matrix captures the statistical distribution of these locations and the ratio of the matrix's eigenvalues tells us the linearity of these points. This gives us a useful estimate of the local curvature. As seen in Figure 7, areas of high curvature have a small eigenvalue ratio whereas more linear edges have a high eigenvalue ratio.

The number of neighbors we include in our estimation will directly impact the perceived curvature. We cannot take into account every possible window of neighbors around each location of the is-

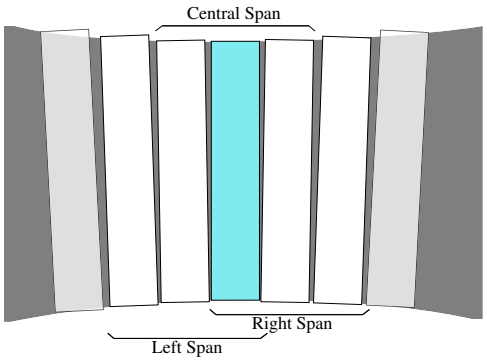


Figure 8: For each bridge strip being evaluated, metrics are computed using three separate windows: Leftward, Centered, and Rightward. These windows are evaluated independently to best represent the ability to place a full bridge at this location.

land border, but we can be smart about our estimation. We already know the size of the island and can therefore compute the combined bridge width needed for stability. This width is directly proportional to the area of the island. Using this width, we have a window over which to compute the curvature. In order to account for local features we compute three separate values: A leftward window, a rightward window, and a centered window. Figure 8 shows an example of this. Of the three windows, we choose the minimum cost to represent the candidate's best available option if it were to be a part of a bridge.

Finally, if we are to approximate a bridge of a particular width by a series of strips, we need to account for the cost of neighboring strips that will also comprise the finished bridge. That is, given a particular strip, how suitable are its fellow neighbors in joining it? We treat this in the same fashion as the curvature, by computing the variance of cost along the island strips for the same sized windows, leftward, rightward, and centered. This value is particular helpful in avoiding outliers which appear to be good bridge locations but can only tolerate a bridge width smaller than necessary to support the island. Later, we can simply expand our bridge across the cheapest neighbor strips available. Using this information, we can also weight the bridges to prefer locations that are directly adjacent to high variability, something that can often be seen in the physical media.

8 Growing the Stencil

With the knowledge of our islands and the cost of bridges available, we can now grow the primary stencil structure by constructing the minimal spanning tree. For every edge chosen for the tree, we unite the two islands into a new structure we refer to as an *island group*. This group has all the properties of an island itself but ignores self connections. When the tree is complete, we will be left with a single island group structure containing all bridge strips we need for the foundation of our stencil. One advantage to this approach is that the number of edges is monotonically increasing so a solution will always be found. Figure 9 shows an example of this process on a trefoil knot.

Once we are satisfied with the locations of each island's bridges, we grow them to the size required to support the stencil area. For any bridge between two islands, the size of the smaller island will be used. This ensures that large islands do not obscure details of smaller ones. It also ensures that the frame of the stencil does not generate excessively large bridges.

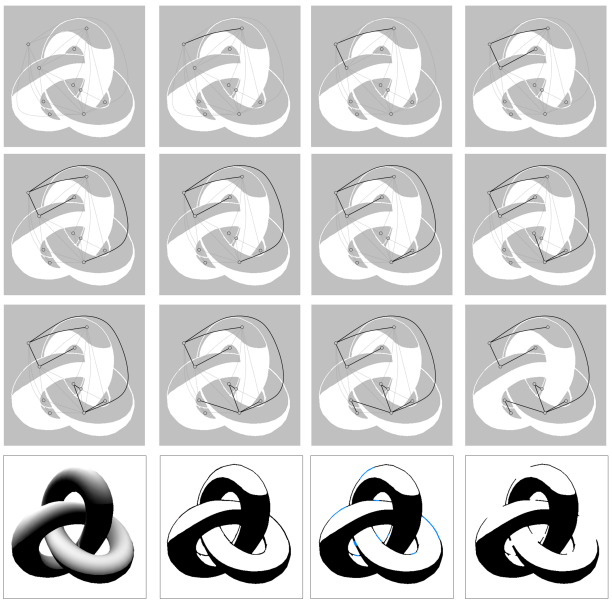


Figure 9: Each iteration of the stencil creation algorithm examines edges which directly connect to the existing spanning tree. Each edge reports its cost as that of the minimum cost bridge strip in its set. The edge with the least costing representative is added to the spanning tree.

To grow a bridge, we start with the one approximator strip chosen as the starting location. Then we iteratively add the neighboring strip with the lowest cost. Once we have achieved the desired width, the strip endpoints can be used as vertices to render a white polygon which will bridge the two islands. If the actual error exceeds the estimated error by a user-defined amount, we will cut the bridge size in half and add an additional support bridge in the next cheapest location. In practice, most stencils do not require extra bridges for added support, even with a flimsy stencil media.

9 Results

Our method creates effective and physically valid stencils when applied to either simple or complex models. The length of time required to generate the stencil will be proportional to the total perimeter of all islands found in the image. Providing proper weights for the metrics of the cost function is a task left to the user, but in practice we found general rankings of importance that seemed to hold true. For simple objects with small bridge lengths, lighting variance appeared to be the most crucial factor in generating high quality stencils. As more detailed silhouettes appeared, local curvature began to trump lighting. Both of these metrics become less important when one bridge length was particularly longer than other choices.

Figure 10 demonstrates the importance of the metrics we use to determine good bridges. Using only intensity difference and bridge length achieves reasonable results, but destroys interesting features like the subtle curves of the pawns base. By raising the weighting of the local curvatures we see the lowest cost bridge move to a more uniform region. Figure 11 shows several more models that have been turned into stencils while Figure 12 shows several stencils that were fabricated for prints and gobos. The painted images were created using two layers, the empty silhouettes as a base with the generated stencil on top. Cutting for prints was done by hand

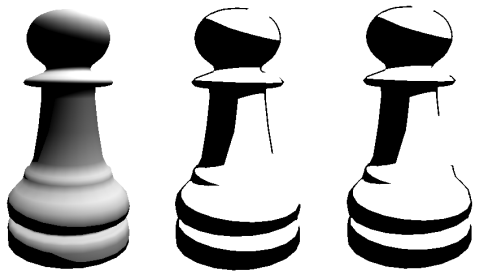


Figure 10: Left: Grayscale rendering of a pawn, Center: Stencil generated when local curvature is ranked relatively low compared to lighting, Right: Stencil generated when curvature is slightly more important than lighting

with a craft knife while the metal gobos were photoetched.

10 Conclusions and Future Work

We present an easy-to-use and powerful method for creating stencils from polygonal models and images. The method allows artistic control over the appearance of the stencil through the view, light position, and lighting threshold, but automates the creation of stencil shapes by connecting the stencil together into a single connected sheet for fabrication and use. We have demonstrated our system on a variety of models, and shown the stencils are robust enough to be used with real media such as painted stencils and theatrical gobos.

One additional step that might improve the structural integrity of the stencils would be to perform a final refinement. This could conceivably be done by measuring the physical forces on each island by a pull such as gravity and attempting to maximally cancel it with minimal added edges.

Most of the methods presented in this paper address the automatic portion of the stencil process, but adding user interaction to drag and drop bridges might also aid in the creation process. Though we feel we have captured the most essential bridge cost metrics, there are many more which could be taken into account. The added statistics will likely have user-preferred weights, and not be inherently more important than one other. We will leave finding additional evaluation criteria to future work.

Finally, scaling is an issue in stencil creation, particularly with respect to aliasing issues stemming primarily from the line drawing. Using a technique for scalable line drawings [Ni et al. 2006] might yield more clearly abstracted results for smaller scaled stencils.

References

DeCarlo, D., and Santella, A. 2002. Stylization and abstraction of photographs. *ACM Trans. Graph. 21*, 3, 769–776.

DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., and Santella, A. 2003. Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3, 848–855.

Eisenberg, M., Elumeze, N., Buechley, L., Blauvelt, G., Hendrix, S., and Eisenberg, A. 2005. The homespun museum: computers, fabrication,and the design of personalized exhibits. In *C&C '05: Proceedings of the 5th conference on Creativity & cognition*, ACM Press, New York, NY, USA, 13–21.

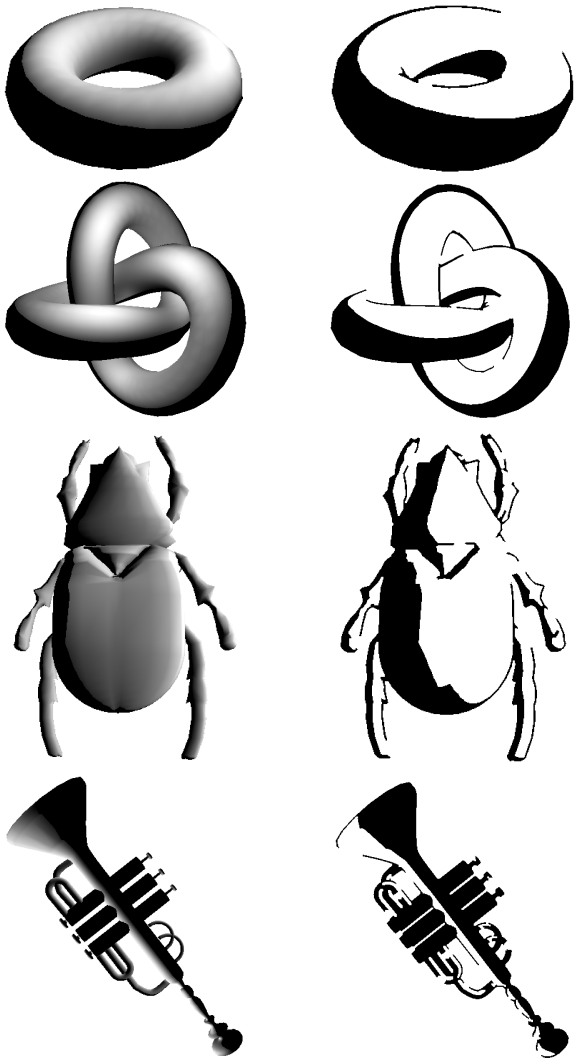


Figure 11: Several different stencils created using our algorithm.

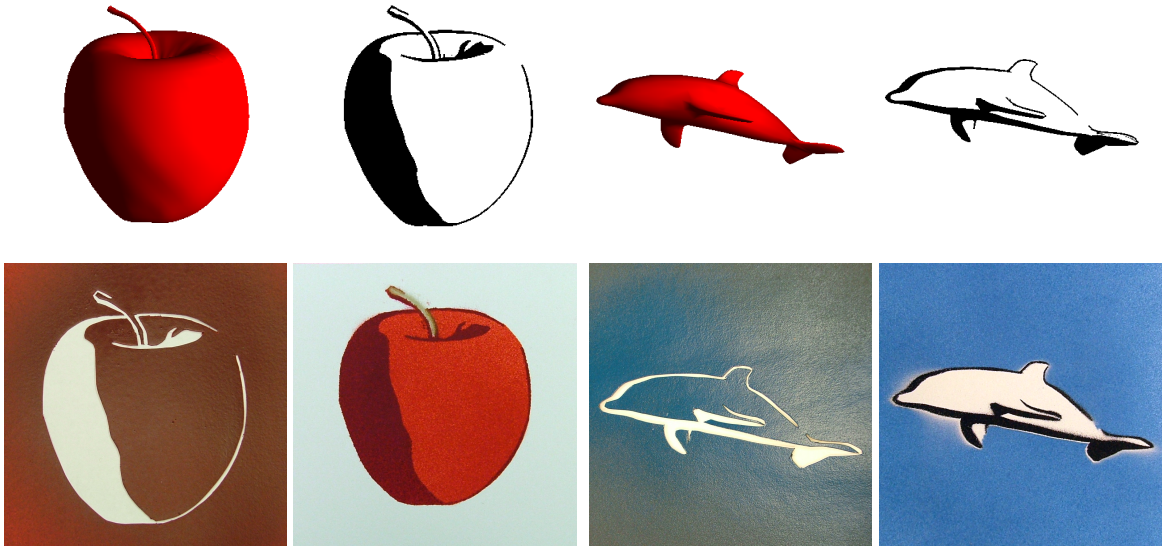
Ferguson, H. 1992. Computer interactive sculpture. In *I3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 109–116.

Girshick, A., Interrante, V., Haker, S., and Lemoine, T. 2000. Line direction matters: an argument for the use of principal directions in 3d line drawings. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, ACM Press, New York, NY, USA, ACM, 43–52.

Hausner, A. 2001. Simulating decorative mosaics. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 573–580.

Isenberg, T., Freudenberg, B., Halper, N., Schlechtweg, S., and Strothotte, T. 2003. A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics and Applications 23*, 4, 28–37.

Kaplan, C. S., and Salesin, D. H. 2004. Islamic star patterns in absolute geometry. *ACM Trans. Graph. 23*, 2, 97–119.



KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 657–664.

LANG, R. J. 1996. A computational algorithm for origami design. In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry*, ACM Press, New York, NY, USA, 98–105.

MARKOSIAN, L., KOWALSKI, M. A., GOLDSTEIN, D., TRYCHIN, S. J., HUGHES, J. F., AND BOURDEV, L. D. 1997. Real-time nonphotorealistic rendering. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, ACM, 415–420.

MITANI, J., AND SUZUKI, H. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA, 259–263.

NI, A., JEONG, K., LEE, S., AND MARKOSIAN, L. 2006. Multi-scale line drawings from 3d meshes. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, ACM, 133–137.

RASKAR, R., AND COHEN, M. 1999. Image precision silhouette edges. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, ACM, 135–140.

RASKAR, R., ZIEGLER, R., AND WILLWACHER, T. 2002. Cartoon dioramas in motion. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, ACM Press, New York, NY, USA, 7–ff.

WILLIAMS, J., AND ROSSIGNAC, J. 2005. Mason: morphological simplification. *Graph. Models 67*, 4, 285–303.

WILLIAMS, J., AND ROSSIGNAC, J. 2005. Tightening: curvature-limiting morphological simplification. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, ACM Press, New York, NY, USA, ACM, 107–112.

Figure 12: The images above show the 4-stage progression of the two stencils, from 3D model, to stencil design, to the fabricated gobo or stencil frame, and finally to the projected shadow and painted image. These stencils were produced using an earlier version of our algorithm.

WOOD, Z., HOPPE, H., DESBRUN, M., AND SCHRÖDER, P. 2004. Removing excess topology from isosurfaces. *ACM Trans. Graph. 23*, 2, 190–208.

WYVILL, B., VAN OVERVELD, K., AND CARPENDALE, S. 2004. Rendering cracks in batik. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, 61–149.

YEN, J., AND SÉQUIN, C. 2001. Escher sphere construction kit. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 95–98.